What is claimed is:

1. A computer system that enables the efficient accessing of Java objects and methods by C++ graphical user interfaces, the computer system comprising:

a processor that runs a software program, wherein the software program generates:

a Java Virtual Machine;

a Java Native Interface ("JNI") boundary; and

a C++ environment, wherein a JNI application programming interface ("API") call across the JNI boundary is required to access the Java Virtual Machine from the C++ environment, the C++ enrivonment comprising:

a graphical user interface, wherein the graphical user interface comprises callback code that is executed to issue one or more method requests; and

a base proxy object, comprising one or more functions that encapsulate one or more JNI API calls necessary to call a Java method in the Java Virtual Machine based on the one or more method requests of the graphical user interface.

2. The computer system of claim 1, wherein the Java Virtual Machine comprises:

a Java object, comprising:

an attribute; and

one or more methods that are executed to enter, retrieve or modify the attribute; and

wherein the base proxy object makes the one or more JNI API calls across the JNI boundary to call the one or more methods of the Java object based on the one or more method requests of the graphical user interface.

3. The computer system of claim 2, wherein the C++ environment further comprises:

a C++ proxy object that proxies the Java object, the C++ proxy object comprising:

one or more methods that corresp[redacted] to the one or more methods of the Java object and that call one or more functions of the base proxy object when executed, wherein the one or more methods of the C++ proxy object are executed in response to the one or more method requests of the graphical user interface.

4. The computer system of claim 3, wherein the C++ graphical user interface executes for a finite length of time and the C++ proxy object and the Java object exist in the C++ environment and the Java virtual machine during the C++ graphical user interface execution.

5. The computer system of claim 3, wherein the Java object is an instance of an instantiated Java class and the C++ proxy object is created as a result of the instantiation of the Java class.

6. The computer system of claim 5, wherein the C++ proxy object includes instance data that identifies the Java object and locates the Java object in the Java virtual machine and wherein the instance data is passed from the Java virtual machine to the C++ proxy object when the C++ proxy object is created.

7. The computer system of claim 3, wherein the C++ proxy object includes one or more method names that name the one or more methods of the Java object and wherein the C++ proxy object passes the one or more method names to the base proxy object when calling the one or more functions of the base proxy object.

8. The computer system of claim 7, wherein one or more method IDs identify the one or more methods of the Java object and the base proxy object retrieves the one or more method IDs using the one or more method names provided by the C++ proxy object.

9. The computer system of claim 8, wherein the base proxy object passes the one or more method IDs to the Java virtual machine when making the one or more JNI API calls across the JNI boundary to call the one or more methods of the Java object.

10. The computer system of claim 8, wherein the base proxy object caches the one or more method IDs in a C++ hash table that is accessible by the C++ proxy objects and the base proxy object.

11. The computer system of claim 2, wherein the Java object is one of the following: a user object, for adding or modifying a user; a node object, for adding or modifying a node; a node group object, for adding or modifying a node group; a tool object, for adding or modifying a tool; and a role object, for adding or modifying a role.

12. The computer system of claim 1, wherein the base proxy object further comprises a mapping mechanism for mapping Java data types to C++ data types.

13. A method for efficient accessing of Java objects and methods by C++ graphical user interfaces, the method comprising:

a C++ graphical user interface issuing a method request to a C++ proxy object;

the C++ proxy object passing method data to a base proxy object based on the method request;

the base proxy object processing the method data; and

a Java object executing a Java method based on the processed method data.

14. The method of claim 13, further comprising, if the executed Java method is a get method, returning a pointer to C++ data.

15. The method of claim 13, wherein the C++ proxy object includes one or more methods and the C++ graphical user interface issuing a method request to a C++ proxy object comprises executing callback code that invokes a C++ proxy object method.

16. The method of claim 13, wherein base proxy object includes one or more functions and the C++ proxy object passing method data to a base proxy object

based on the method request comprises processing the method request and calling a base proxy object function, wherein the base proxy object function call includes method data.

17. The method of claim 16, wherein the base proxy object processing the method data comprises:

executing the called base proxy object function;

getting a method ID based on the method data; and

issuing JNI API calls with the method ID to call the Java method.

18. The method of claim 13, further comprising:

obtaining the Java object via a JNI API call, wherein the Java object instance data is passed through a JNI; and

initiating C++ proxy object linkage to the Java object, wherein the Java object instance data is used to create the C++ proxy object.

19. A computer readable medium containing instructions for enabling the efficient accessing of Java objects and methods by non-Java graphical user interfaces, by:

a non-Java graphical user interface issuing a method request to a non-Java proxy object;

the non-Java proxy object passing method data to a base proxy object based on the method request;

the base proxy object processing the method data; and

a Java object executing a Java method based on the processed method data.

20. The computer readable medium of claim 19, wherein the non-Java graphical user interfaces are C++ graphical user interfaces.